In re Application of Winkelman, *et al.*
Application No. 09/593,973
Reply to Office Action
Page 2

REMARKS

This Reply is submitted in response to the Office Action dated October 24, 2005.

Withdrawal of the Previous Rejections

The Applicant acknowledges the apparent withdrawal of the previous rejections of claims 1, 3, and 4 under Section 102 based on U.S. Patent 5,835,712 issued to *DuFresne*, as well as the withdrawal of the previous rejections of claims 2 and 5-15 under Section 103 based on *DuFresne* and U.S. Patent 6,394,354 issued to *Wilz, Sr., et al.*

Claim Rejections – 35 U.S.C. § 102

The Examiner rejected claims 1, 3, 4, 7-9, 12-14, 16, 18, 19, 22-24, 26, and 27 under Section 102(e), as being anticipated by U.S. Patent 6,031,623A issued to *Smith, et al.* (the "*Smith*" patent). The rejected claims include independent claims 1, 3, and 19.

As provided in Section 2131 of the MPEP: "A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described in a single prior art reference." *Verdegaal Bros. v. Union Oil Co.*, 814 F.2d 628, 631, 2 U.S.P.Q.2d 1051, 1053 (Fed. Cir. 1987). "The identical invention must be shown in as complete detail as is contained in the ... claim." *Richardson v. Suzuki Motor Co.*, 868 F.2d 1226, 1236, 9 U.S.P.Q.2d 1913, 1920 (Fed. Cir. 1989). The elements must be arranged as required by the claim.

Independent claim 19 is not anticipated because *Smith* does not disclose or describe every limitation and element recited in the claim. *Smith* discloses a software system using object-oriented programming, which is a procedural language. In contrast, the system recited in claim 19 involves a simple, declarative language. The object-oriented programming described in *Smith* includes class definitions, hierarchies, run-time polymorphism, pointer variables, virtual function tables, and binding steps. (*Smith*, col. 10, lines 20-40). None of those elements are recited in the pending claims.

Smith does not disclose "a recipe text file comprising a plurality of formatting descriptions." *Smith* does not disclose or describe a recipe text file, either expressly or inherently. The Examiner suggests that the "class definition [in *Smith*] corresponds to a recipe

In re Application of Winkelman, *et al.*
Application No. 09/593,973
Reply to Office Action
Page 3

text file." (Office Action, p. 3). The class definition in *Smith* is not identical to the recipe text file recited in the claims, as required to support a 102 rejection. The recipe text file is a plain text file (Application, Fig. 4) built using simple BNF grammar. (Application, p. 12, line 17). In contrast, a class definition in object-oriented programming defines methods; each method may include a variety of functions to be executed. (*Smith*, col. 10, lines 20-40). *Smith* does not disclose the use of a plain-text file identical to the recipe text file recited in the claims.

Smith does not disclose "a link to one or more of a plurality of name/value pairs." The Examiner suggests that the "pointers at least correspond to links." (Office Action, p. 3). The pointer variables in *Smith* are not identical to the links recited in the claims. The link recited in the claims is a simple keyword or "link name" (Application, p. 9, lines 16-18) that is used to connect the formatting description to the content. In contrast, in object-oriented programming, the pointer variables stored in a virtual function table in *Smith* are used to control the values of objects at run time. (*Smith*, col. 10, lines 20-40). *Smith* does not disclose the use of a link identical to the link recited in the claims.

Smith does not disclose "a plurality of name/value pairs stored in an input data file." The Examiner suggests in the Office Action that the "values determined at run time [correspond] to an input data file." (Office Action, p. 3). The run-time values in *Smith* are not identical to the input data file as recited in the claims. The input data file recited in the claims is a simple, declarative list of names and values (Application, Fig. 3). In contrast, in object-oriented programming, the values determined at run time are controlled or manipulated by the virtual function tables and pointer variables. (*Smith*, col. 10, lines 20-40). *Smith* does not disclose an input data file identical to the one recited in the claims.

Smith also does not disclose "an execution module" as recited in the claims. The Examiner suggests in the Office Action that the "binding of a class definition to a particular run-time instantiation corresponds to [the] conversion to a sequence of executable objects [by the execution module recited in the claims]." (Office Action, p. 4). The binding of a class definition in *Smith* is not identical to the conversion step recited in the claims. Moreover, the "executable

In re Application of Winkelman, *et al.*
Application No. 09/593,973
Reply to Office Action
Page 4

objects" recited in the claims do <u>not</u> correspond to the "objects" referred to in the context of object-oriented programming. As recited in claim 19, for example, the "sequence of executable objects" is executed "to render one or more of said plurality of name/value pairs into said output." The rendering process is described on pages 17 and 18. *Smith* does not disclose an execution module that converts a recipe text file into a sequence of executable objects which, when executed, render one or more of a plurality of name/value pairs into an output, as recited in the claims.

As to claim 26, the Examiner suggests in the Office Action that the "polymorphic class definitions and/or the methods evolved from legacy software, as noted above, correspond to modification of the input data text file of a base method." (Office Action, p. 4). *Smith* does not disclose an execution module "further configured to modify said input data text file to include one or more modified name/value pairs" as recited in dependent claim 26. As addressed above, none of the elements in *Smith* are identical to the input data text file recited in the claims. Accordingly, none of the steps *Smith* for modifying the various elements in *Smith* could possibly be identical to the module recited in claim 26 for modifying the input data text file.

Because *Smith* does not disclose or describe a program or product that is identical to the product recited in claim 19, including every limitation or element, the 102 rejection is unsupported by the art and should be withdrawn. The rejection of dependent claims 22-24, 26, and 27 should also be withdrawn for the same reason.

<u>Independent claim 1</u> is not anticipated because *Smith* does not disclose or describe a system that is identical to the system recited in claim 1, including every limitation or element. Accordingly, the 102 rejection of claim 1 is unsupported by the art and should be withdrawn. The rejection of dependent claims 12-14, 16, and 18 should also be withdrawn.

<u>Independent claim 3</u> is not anticipated because *Smith* does not disclose or describe a method that is identical to the system recited in claim 3, including every limitation or element. Accordingly, the 102 rejection of claim 3 is unsupported by the art and should be withdrawn. The rejection of dependent claims 4, and 7-9 should also be withdrawn.

In re Application of Winkelman, *et al.*
Application No. 09/593,973
Reply to Office Action
Page 5

<u>Claim Rejections – 35 U.S.C. § 103(a)</u>

The Examiner rejected claims 2, 5, 6, 10, 11, 15, 17, 20, 21, 25, and 28 under Section 103(a) as being obvious and unpatentable over *Smith*. The 103 rejection is based on *Smith* alone, or in combination with "official notice." These rejected claims includes dependent claims only.

<u>MPEP § 2142</u>: A *prima facie* case of obviousness requires that: (1) the prior art references teach or suggest all of the features of the claimed invention; (2) there is some suggestion or motivation to modify or combine the prior art references; and, (3) there is a reasonable expectation of success in combining the prior art references. MPEP § 2142; *In re Vaeck*, 947 F.2d 488 (Fed. Cir. 1991). "The examiner bears the initial burden of factually supporting any *prima facie* conclusion of obviousness. If the examiner does not produce a *prima facie* case, the applicant is under no obligation to submit evidence of non-obviousness." MPEP § 2142; *In re Geiger*, 815 F.2d 686, 690 (Fed. Cir. 1987).

The rejected claims 2, 5, 6, 10, 11, 15, 17, 20, 21, 25, and 28 depend from one of the independent claims, 1, 3, or 19. As discussed above, the independent claims are not anticipated by *Smith* for a variety of reasons. In addition, the *Smith* patent (either alone or in combination with any other reference) does not teach or suggest all the features of the embodiments claimed in the independent claims or the dependent claims.

Smith does not teach or suggest "a recipe text file comprising a plurality of formatting descriptions." The class definition in *Smith* does not teach or suggest the recipe text file recited in the claims. The recipe text file is a plain text file (Application, Fig. 4). In contrast, a class definition in object-oriented programming defines methods; each method may include a variety of functions to be executed. (*Smith*, col. 10, lines 20-40). *Smith* does not teach or suggest the use of a plain-text file like the recipe text file recited in the claims.

Smith does not teach or suggest "a link to one or more of a plurality of name/value pairs." The pointer variables in *Smith* do not teach or suggest the links recited in the claims. The link recited in the claims is a simple keyword or "link name" (Application, p. 9, lines 16-18) that is used to connect the formatting description to the matching content. In contrast, in object-

In re Application of Winkelman, *et al.*
Application No. 09/593,973
Reply to Office Action
Page 6

oriented programming, the pointer variables stored in a virtual function table in *Smith* are used to control the values of objects at run time. (*Smith*, col. 10, lines 20-40). *Smith* does not teach or suggest the use of a link such as the one recited in the claims.

Smith does not teach or suggest "a plurality of name/value pairs stored in an input data file." The run-time values in *Smith* do not teach or suggest the input data file recited in the claims. The input data file recited in the claims is a simple, declarative list of names and values (Application, Fig. 3). In contrast, in object-oriented programming, the values determined at run time are controlled or manipulated by the virtual function tables and pointer variables. (*Smith*, col. 10, lines 20-40). *Smith* does not teach or suggest an input data file such as the one recited in the claims.

Smith also does not disclose "an execution module" as recited in the claims. The binding of a class definition in *Smith* does not teach or suggest the conversion step recited in the claims. The "executable objects" recited in the claims do <u>not</u> correspond to the "objects" referred to in the context of object-oriented programming. As recited in claim 19, for example, the "sequence of executable objects" is executed "to render one or more of said plurality of name/value pairs into said output." The rendering process is described on pages 17 and 18. *Smith* does not teach or suggest an execution module that converts a recipe text file into a sequence of executable objects which, when executed, render one or more of a plurality of name/value pairs into an output, as recited in the claims.

All claim limitations must be considered, especially when missing from the cited art. The object-oriented programming described in *Smith* is a procedural language that does not teach or suggest any features or elements that might be used in a software system such as the one recited in claim 19, for example, which involves a simple, declarative language. The highly complex and hierarchical procedural elements from *Smith* simply do not translate or apply to a system with simple, declarative elements. In practice, adopting the procedural approach in *Smith* would "teach away" from the concept of using declarative elements. Because *Smith* does not teach or suggest all the limitations of the independent claims 19, 1 and 3, it also does not teach or suggest

In re Application of Winkelman, *et al.*
Application No. 09/593,973
Reply to Office Action
Page 7

all the limitations of the dependent claims. The rejection of the dependent claims 2, 5, 6, 10, 11, 15, 17, 20, 21, 25, and 28 is not supported by the art and should be withdrawn.
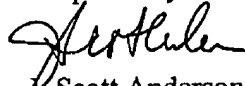
CONCLUSION

Claims 1-28 remain pending in the application. In light of this reply and the remarks presented, the Applicant respectfully submits that all the claims of the application are patentable and in condition for allowance.

The undersigned is available at (404) 881-7821 if the Examiner has any questions or requests that may be resolved by telephone in order to expedite the examination of this application.

The Applicants submit herewith a Petition and Fee for a One-Month Extension of Time. The Applicants do not believe any other requests for extension of time or fees are required, beyond those provided for in documents accompanying this paper. In the event, however, that additional extensions of time are necessary to allow the consideration of this paper, such extensions are hereby petitioned-for under 37 C.F.R. § 1.136(a) and any fee required therefor (including fees for net addition of claims) is hereby authorized to be charged to Deposit Account No. 16-0605.
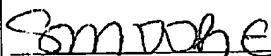
Respectfully submitted,

V. Scott Anderson
Registration No. 48,563

Customer No. 00826
ALSTON & BIRD LLP
Bank of America Plaza
101 South Tryon Street, Suite 4000
Charlotte, NC 28280-4000
Atlanta Office (404) 881-7000
Fax (404) 881-7777

CERTIFICATE OF FAX TRANSMISSION

I hereby certify that this paper is being transmitted by facsimile to (571) 273-8300 at the U.S. Patent and Trademark Office on this, the 24 day of February, 2006.

Shana Moore

ATLo1/12155290v2